

Guten Morgen, ich begrüße Sie zu dem Vortrag 'Foswiki, Padre und OTRS - Erfolgreich Software entwickeln mit Perl'. Ich möchte Ihnen die Programmiersprache Perl vorstellen und an Hand der drei Anwendungen erläutern, wie man mit Perl erfolgreich Software entwickeln kann.

Damit Sie wissen, mit wem Sie es hier zu tun haben, eine kurze Vorstellung von mir.

Mein Name ist Renée Bäcker und bin selbständiger Perl-Programmierer. Seit 2007 bringe ich das deutschsprachige Perl-Magazin "\$foo" heraus. Ich habe also den ganzen Tag fast nur mit Perl zu tun. Hier auf der CeBIT bin ich die meiste Zeit am Stand der Perl Foundation in der Open Source Project Lounge zu finden.

Als erstes werde ich Perl ganz Allgemein vorstellen. Ich werde während des Vortrags keine Perl-Befehle oder ähnliches zeigen. Das würde zu weit führen und es besteht die Möglichkeit, darüber an unserem Stand zu diskutieren. Danach werde ich die drei Open Source Projekte Foswiki, Padre und OTRS kurz vorstellen. Diese drei Projekte sind in Perl geschrieben und sehr erfolgreich. Zum Schluss werde ich noch auf einige Punkte eingehen, wie man effektiv Projekte mit Perl umsetzen kann.

Hört man sich bei Managern bzw. IT-Verantwortlichen um, so wird häufig *nicht* Perl genannt bei der Frage nach den bekannten Programmiersprachen. Hier fallen häufiger die Namen PHP, Python, Java und Ruby on Rails...

... was aber nicht bedeutet, dass Perl nirgends eingesetzt wird. Perl wird häufig als veraltet bezeichnet, aber nur von denjenigen, die Perl vor Jahren aus den Augen verloren haben. Was richtig ist ist, dass Perl schon länger existiert. Die erste Version von Perl ist schon 23 Jahre alt und Perl 5 ist immerhin schon 16 Jahre alt. Aber Perl 5.11.5 - eine reine Entwicklerversion - ist gerade einmal 11 Tage alt.

Um die Installation muss man sich auf den meisten Unix/Linux-System nicht kümmern, weil es automatisch mit ausgeliefert wird. Und für Windows gibt es mittlerweile zwei große Alternativen: ActivePerl von ActiveState und ein reines OpenSource-Bundle mit dem Namen StrawberryPerl.

Durch die große Verfügbarkeit wird der Einsatz von Perl häufig gar nicht richtig wahrgenommen - fragen Sie doch mal ihren Systemadministrator. Auch Jobs, in den Perl eine Voraussetzung ist, gibt es genug.

Doch jetzt zu Perl: Was ist Perl eigentlich? Wofür kann man Perl verwenden? Was sind die Stärken von Perl und was die Schwächen?

Perl ist eine dynamische Programmiersprache, die sehr stabil ist. Diese Stabilität zeigt sich in den vielen Tests. Perl wird auf vielen Plattformen in den unterschiedlichsten Versionen mit den unterschiedlichsten Einsatzbereichen eingesetzt. Auf die Tests von Perl komme ich später noch. Im Laufe der Zeit wurde Perl immer ausgereifter. Durch viele Konstrukte, die sehr mächtig sind, erhält man einen sehr kompakten Code. Unter Perl-Programmierern sagt man, dass man in Perl das in *einer* Zeile machen kann, wofür man in Java mindestens 15 Zeilen Code benötigt.

Perl und viele Erweiterungen stehen unter der Artistic License, die es auch erlaubt, kommerzielle Anwendungen damit umzusetzen.

Perl trifft man häufig bei Webanwendungen...

z.B. XING. XING wurde von Anfang an in Perl entwickelt. Mittlerweile sind ein paar Teile der

Oberfläche mit Ruby und Rails umgesetzt, aber der Großteil wird immer noch in Perl gemacht.

Auch die Anwendung "OTRS", über die ich gleich noch mehr sagen werde, ist komplett in Perl geschrieben.

Auch Foswiki gehört zu den Webanwendungen, die mit Perl umgesetzt sind.

Die Auflistung könnte man hier fast unendlich weitermachen...

Wie ich vorhin schon sagte, wird Perl sehr häufig in der Systemadministration eingesetzt, um bestimmte Abläufe zu vereinfachen.

Überwachung von Dateisystemen und Prozessen

Benutzer-Administration

Name Services

Datenbankverwaltung

Verzeichnisdienstverwaltung

Auswertung von Log-Dateien, Sicherheit und Netzwerküberwachung

Und für Grafische Oberflächen gibt es mehrere Module in Perl. Die beiden folgenden Screenshots wurden mit wxPerl umgesetzt.

Padre ist so ein Beispiel für Perl bei Grafischen Oberflächen.

Oder hier ein Simulationsprogramm für Brandmeldeanlagen, das man in der Feuerwehr-Ausbildung einsetzen kann.

Das Einsatzgebiet für Perl ist nahezu unbegrenzt... In der Bioinformatik wird Perl sehr häufig auf Grund seiner Stärken in der Textverarbeitung eingesetzt. Durch die Integration von Perl in den Apache mit mod_perl kann man nicht nur Webanwendungen schneller machen, man kann auch den Apachen für komplett andere Server-Aufgaben verwenden, da man mit mod_perl jeden Schritt einer Anfrage manipulieren kann. Allgemein wird Perl sehr häufig eingesetzt, wenn es darum geht, Abläufe zu automatisieren. Auch für die Steuerung von Plasmaschneidern habe ich Perl schon eingesetzt.

Diese Grafik zeigt, dass rund 18% der neuen Open Source Projekte im Jahr 2008 mit Perl umgesetzt wurden. Bei der Untersuchung wurden von Black Duck Software rund 17.000 Open Source Projekte unter die Lupe genommen. Der Löwenanteil setzt immer noch auf C.

Zu den großen Stärken von gehört CPAN, ein riesen Archiv für Perl-Erweiterungen. In der Perl-Gemeinde sind "Tests" weitverbreitet: Auf CPAN gibt es hunderte Erweiterungen zum Thema "Test". CPAN-Module haben in der Regel Tests, mit denen man prüfen kann, ob das Modul auf dem eigenen Rechner wie gewünscht/dokumentiert funktioniert. Und jetzt noch zwei Zahlen, die zeigen, dass die Entwickler von Perl großen Wert darauf legen, dass das Verhalten von Perl stabil ist: Allein der Perl-Kern hat rund 93.000 Tests dazu kommen noch rund 102.000 Tests für die Module, die zu der Standard-Distribution von Perl mitgeliefert werden. Das bedeutet, dass knapp 200.000 Tests durchlaufen wenn Sie Perl aus den Quellen auf ihrem Rechner installieren. Das kann zwar eine Weile dauern, aber dafür wissen Sie auch, dass Perl funktioniert.

Auch die Gemeinschaft rund um die Programmiersprache gehört zu den Stärken. Von vielen hört man, dass die Perl-Gemeinschaft sehr hilfsbereit ist. Und durch die vielen Perl-Veranstaltungen gibt es so etwas wie eine große Familie, zu der aber jeder herzlich eingeladen ist. In vielen Städten gibt

es die sogenannten Perlmongers, die Usergruppen zu Perl.

Durch seine Wurzeln - die Unix-Programme sed, awk, etc... und durch Dinge wie Reguläre Ausdrücke... - hat Perl große Stärken in der Textverarbeitung.

Perl wird auch sehr häufig für die Erstellung von Prototypen verwendet. Man sehr schnell zu Ergebnissen kommen.

Aber wie alles, hat auch Perl gewisse Schwächen: Für Echtzeitsysteme ist Perl nicht so gut geeignet. Auch für die Erstellung von 3D-Spielen ist Perl nicht unbedingt die erste Wahl. Aber hier gibt es gerade Bewegung in ein paar Perl-Erweiterungen. Threads funktionieren "irgendwie", aber die Perl-Threads sind nicht das, was viele von Threads erwarten.

Viele sehen einen großen Nachteil von Perl: Das Deployment von Perl-Webanwendungen ist nicht so einfach wie bei PHP-Webanwendungen, weil es mehr Aktion vom User verlangt. Nur mit "kopieren und loslegen" ist es nicht getan. Für viele Anwender mit einfachem "Shared Webhosting" ist das eine Hürde.

Zu den Schwächen von Perl gehört es auch, dass es oft ganz einfach unterschätzt wird und "Mythen" in Umlauf gebracht werden...

Perl-Code ist nicht lesbar. Klar kann man so programmieren...

... aber das geht auch in anderen Sprachen...

Wer aber in Projekten mitarbeitet, wird solchen Code ausgetrieben bekommen. Dazu gibt es "Best Practices" und für Perl gibt es ein Buch darüber. Diese "Best Practices" können mit dem Modul "Perl::Critic" durchgesetzt werden.

Und für "schönen" Perl-Code gibt es noch Perl::Tidy, womit man einheitlichen Programmierstil erzwingen kann.

So viel zu einem Überblick zu Perl. Kommen wir zu den drei Projekten Foswiki, Padre und OTRS.

Foswiki ist eine Fork von TWiki und gehört zu den beliebtesten Open Source Firmenwiki-Systemen. Foswiki steht für "Free and Open Source Wiki". Die größten Konkurrenten kommen aus dem Closed Source Bereich. Zur Zeit bringt es Foswiki auf ca. 60.000 Installationen in 130 Ländern. Die Anwendung ist in 14 Sprachen verfügbar.

Im Gegensatz zu den meisten anderen bekannten Wikisystemen unterteilt Foswiki die Inhalte in Webs, Topics und Attachments. Ein Web ist dabei eine Sammlung von Topics, die wiederum ein Abschnitt oder ein Dokument sind. Das mit den Webs sehe ich als einen großen Vorteil von Foswiki gegenüber anderen Systemen. Wir haben z.B. für unterschiedliche Projekte verschiedene Webs. Alles in einer einzigen Foswiki-Instanz. Für jedes Web können auch extra Zugriffsrechte vergeben werden, so dass Kunde A Zugriff auf Web X hat - aber keine anderen und Kunde B hat Zugriff auf Web Y.

Das Foswiki ist ideal als Tool für die Zusammenarbeit in Teams. Es ist ein Wiki, so dass Wissen geteilt werden kann. Für Unternehmen ist es oft auch wichtig, dass sich die Tools Infos aus dem LDAP besorgen können, so dass Benutzerinformationen nicht an zig Stellen verwaltet werden müssen. Wie eben schon angedeutet, liegen die Daten strukturiert vor. Und das Foswiki hat viele Plugins - und man kann relativ leicht selbst welche schreiben.

Hier nochmal als Zusammenfassung...

Hier ein paar Screenshots von Foswiki. Das ist die Startseite von Foswiki.org . Man sieht die verschiedenen Webs.

Und mit Plugins kann man aus Daten, die im Wiki hinterlegt sind, auch Grafiken erzeugen. So können sich Chefs schnell die gewünschten Kennzahlen ziehen.

Auch Präsentationen können in dem Wiki direkt hinterlegt werden.

Padre ist ein gutes Beispiel für eine GUI, die mit Perl entwickelt wurde. Es ist eine IDE, mit sehr guter Unterstützung für Perl. Aber auch andere Sprachen werden unterstützt.

Begonnen wurde das Projekt 2008 und die Entwicklung geht sehr schnell weiter. Es werden regelmäßig Releases veröffentlicht.

Die Funktionalität von Padre kann sehr gut über Plugins erweitert werden. Zur Zeit gibt es 56 Plugins für alle möglichen Aufgaben wie die Anbindung an SVN, Refactoring von Perl-Programmen oder auch für Autodia.

Hier ein ziemlich aktueller Screenshot von Padre. Padre hat die meisten Features, die man von einer IDE erwarten würde, wie z.B. den Debugger eingebaut. Unterstützung für "Projekte", Klassenbrowser und vieles mehr.

Wer Fragen zu Padre hat, kann gerne zu unserem Stand kommen. Über die CeBIT-Zeit sind drei Padre-Entwickler dort anzutreffen.

Das dritte Projekt, das ich hier kurz vorstellen möchte, ist OTRS. OTRS steht für "Open Source Ticket Request System". Das System ist bei 60% der DAX-Unternehmen im Einsatz.

Der Name deutet darauf hin, dass es ein reines Ticket-System ist, aber es ist mehr. Es ist ein Framework, das sehr leicht erweitert und auf Unternehmensprozesse angepasst werden kann. Als Erweiterungen gibt OTRS::ITSM mit CMDB, Incident Management und mehr. Für das BSI wurde SIRIOS entwickelt, um CERT-Warmmeldungen erstellen und managen zu können.

Für OTRS gibt es auch noch Pakete wie das FAQ-Modul, TimeAccounting und Webmail

Insgesamt gibt es rund 70.000 Installationen weltweit und OTRS gibt es für 27 Sprachen. Zu den bekanntesten Installationen dürften diese bei NASA, Wikipedia und mymuesli gehören.

Das zeigt, dass OTRS nicht nur in kleinen Unternehmen eingesetzt wird, sondern auch für große Organisationen geeignet ist.

Hier mal zwei Screenshots...

... die die Oberfläche von OTRS zeigen.

Morgen gibt's hier auf der LPI-Bühne um 16:00 Uhr einen Vortrag von Jens Bothe über ITIL-konformes OTRS.

Und hier in der Halle gibt es drei Stände von Unternehmen/Organisationen, die sich mit OTRS gut

auskennen.

Aber was macht diese Projekte so erfolgreich? Sicherlich gibt es viele verschiedene Erfolgsfaktoren für Projekte wie diese. Aber diese hier spielen ganz sicher eine Rolle...

Perl... Über Perl habe ich vorhin ja schon einiges erzählt. Die Stärken von Perl spielen ganz sicher eine Rolle für den Erfolg dieser Anwendung.

Auf die anderen Punkte - CPAN, Plugins und Gute Community - werde ich jetzt noch ein wenig eingehen.

Wie vorhin schon erwähnt ist das CPAN ein Riesenarchiv von Perl-Erweiterungen. Das ganze ist unter cpan.org erreichbar. Zur Zeit gibt es rund 20.000 Perl-Module in dem Archiv, von rund 8.000 Autoren. Da die Autoren aus den unterschiedlichsten Bereichen sind, gibt es mittlerweile für fast jede Aufgabe ein Modul. Möchte man ein Problem lösen, lohnt es sich auf jeden Fall, mal einen Blick auf das CPAN zu werfen.

Viele der Module sorgen auch dafür, dass viele Kritikpunkte entkräftet werden können. Perl 5 hat keine astreine Objektorientierung wie man es aus anderen Sprachen kennt. Die normale Objektorientierung funktioniert zwar und man gewöhnt sich daran, aber in anderen Sprachen gibt es viele Sachen, die die Objektorientierung vereinfachen. Auf dem CPAN gibt es das Modul, das Objektorientierung in Reinform auch für Perl 5 ermöglicht. Und das Catalyst ist an Ruby on Rails angelehnt und das Standard-Webframework im Moment.

In den letzten Jahren hat sich eine "Modern Perl"-Bewegung formiert, die gewisse "Best Practices" umsetzt und sich für "guten" Perl-Code einsetzt.

Für Projekte wie die drei genannten, ist das CPAN natürlich eine wahre Fundgrube an Funktionalitäten. Warum soll ich mich selbst um das Verschlüsseln von Daten kümmern, wenn es dafür schon fertige Bibliotheken gibt? Oder warum sollte ich mich mit dem Excel-Format auseinandersetzen, wenn es schon Module gibt, mit denen ich Excel-Dateien erstellen kann?

Auch für die Autoren bietet das CPAN einige tolle Sachen: Für jedes Modul gibt es automatischen einen Bugtracker. Und man kann sich Namensräume reservieren. Mit aussagekräftigen Namensräumen kann man schon am Modulnamen erkennen, wofür es gut ist.

Für jedes Modul gibt es auch ein Forum auf cpanforum.com

Und bei den CPAN-Testers zeigt sich wieder die "Test"-Kultur. Einige Personen stellen Maschinen mit unterschiedlichem Betriebssystem (insgesamt ca. 60 Plattformen) und unterschiedlichen Perl-Versionen (insgesamt ca. 20 Versionen) zur Verfügung, die immer wieder die aktuellen Module vom CPAN herunterladen und die Installation testen. Im Archiv der CPAN-Tester gibt es einige Millionen Test-Berichte. Nicht jeder Autor hat die ganzen Systeme zur Verfügung und dank der CPAN-Tester weiß man, auf welchen Systemen sein Modul läuft und auf welchen nicht.

Die drei vorgestellten Projekte haben alle ein Plugin-System. Zwar alle unterschiedlich, aber Plugins sind für solche Systeme wichtig. Sie sorgen dafür, dass das System leicht erweiterbar ist. Außerdem rekrutieren sich viele Entwickler daraus, dass sie anfangs nur Plugins haben wollten, dann selbst entwickelt haben und dann das Produkt selbst weiterentwickelt haben. Je mehr unterschiedliche Plugins es gibt, umso mehr Bereiche können abgedeckt werden.

Für Open Source Projekte ist das wichtiger als für Closed Source Projekte, aber die Gemeinschaft,

die hinter einem Projekt steht, ist enorm wichtig. Was nützt mir ein Projekt, bei dem ich niemanden fragen kann, wenn etwas nicht funktioniert? Für alle hier genannten Projekte gibt es Mailinglisten, Foren, IRC-Channels und vieles mehr, in denen man Fragen loswerden kann. Eine weitverteilte Gemeinschaft hilft auch dabei, einen 24/7-Support zu haben, ohne dass man jemanden dafür bezahlen muss. Eine freundliche Nutzer-/Entwicklergemeinschaft lädt auch dazu ein, sich etwas mehr mit dem Projekt zu beschäftigen.

Die genannten Sachen sind nicht die einzigen Punkte, die zu einem erfolgreichen Projekt beitragen, aber sie sind wichtig. Aber wie kann ich eine Software erfolgreich mit Perl erstellen? Da hätten wir Module, "Best Practices", Tests, Devel::Cover, Devel::NYTProf

Software sollte in kleinen Bausteinen erstellt werden. Das hilft ungemein, wenn man später einen kleinen Teil ersetzen muss oder wenn man auf der Suche nach einem Fehler ist. In eigenen Modulen sollte man die Funktionalitäten zusammenfassen, die häufiger benötigt werden. Diese Module sollte man idealerweise im CPAN-Stil erstellen. Auch hierzu gibt es Hilfsmodule, die die richtige Verzeichnisstruktur und ein paar erste Basis-Tests erstellen.

Aber bevor man sich die Arbeit macht, etwas selbst zu implementieren, kann man ja mal auf CPAN vorbeischaun, ob nicht schon jemand die Arbeit gemacht hat und auf CPAN veröffentlicht hat.

Ein ganz wichtiger Teil bei guten Projekten ist, dass man sich auf "Best Practices" einigt. Damian Conway hat seine Erfahrungen in dem Buch "Perl Best Practices" zusammengefasst. Mittlerweile basieren viele Programmierrichtlinien in Unternehmen auf diesen Erfahrungen. Diese "Best Practices" sorgen auch dafür, dass der Code lesbar bleibt und alle Entwickler die gleichen Sachen nutzen. Conway hat mal gesagt, dass man so programmieren sollte, als wäre der Nachfolger ein Psychopath, der weiß wo man wohnt...

Diese Regeln von Conway wurden in das Modul Perl::Critic eingebaut. Damit regelmäßig den Code überprüfen lassen und dann bekommt man alle Entwickler dazu, die gleichen Sachen zu machen. Das Modul, überprüft den Code auf bestimmte Perl-Konstrukte.

Wer das erstmal ausprobieren möchte, kann seinen Quellcode mal auf perlcritic.com überprüfen lassen.

Tests, Tests, Tests... Tests zu schreiben gehören nicht gerade zu den Lieblingsaufgaben eines Programmierers, aber man kann sich viel Ärger damit ersparen. Auf CPAN gibt es rund 450 Module, die sich nur mit Tests beschäftigen. Damit kann man die meisten Test-Fälle berücksichtigen.

Ein Test ist besser als keiner. Man muss ja nicht gleich mit 1000 Tests anfangen. Aber dann wenn Bugs auftreten, sollte man mit dem Test-Schreiben anfangen. Hat man ein Projekt erstmal ohne Tests angefangen, ist es schwierig, noch welche einzuführen.

Wenn man die Tests geschrieben hat, kann man dann testen, ob man auch wirklich alle Codestellen überprüft hat. In einer netten HTML-Ausgabe sieht man, welche Codestellen getestet wurden und welche nicht.

Sollte man auf Performance-Probleme treffen, kann man mit Devel::NYTProf prüfen, welche Codeteile langsam sind und welche nicht. Häufig würde man an der falschen Stelle optimieren. Die Ausgabe ist sehr ähnlich zu dem von Devel::Cover.

Das waren ein paar Punkte, die man bei der Softwareentwicklung mit Perl berücksichtigen sollte.

Ich möchte Sie noch dazu einladen, bei unserem Stand in der Open Source Project Lounge vorbeizuschauen. Heute ist der Schwerpunkt "Padre", morgen haben wir dann ein paar Entwickler von "Foswiki" vor Ort und am Freitag liegt der Schwerpunkt bei "OTRS". Aber natürlich können wir auch über andere Themen sprechen...

Ich bedanke mich für Ihre Aufmerksamkeit